# Finding the closest pair of points

2D problem: Given an array $P$ of $n$ points in the plane, find the closest pair. Assume that the distance between two points is given by the Euclidian distance.

## Questions

1. Formulate the 1D version of the closest pair. How can you solve it, and how fast? Try to extend this solution to the 2D problem: does it work?

   For the remaining problems we consider the 2D version.

2. Describe how you can find a vertical line that splits $P$ in half. How long does this take?

3. Consider a point $p \in P$. Prove that in order for a point $q$ to be within distance $d$ from $p$, then both the horizontal and vertical distance between $p$ and $q$ must be smaller than $d$.

4. With the notation in the slides, show an example of points where the strip of width $d$ around the middle vertical line may contain $\Omega(n)$ points. What does this mean for the running time of the whole algorithms? Write a recurrence.

5. Let $P$ be an array of $n$ points in the plane. Assume you are given an array $P^X$ containing the points in $P$ sorted by their x-coordinates, and an array $P^Y$ containing the points in $P$ sorted by their y-coordinates.

   Note that the vertical line that splits $P$ in half is the median of the x-coordinates of the points in $P$, which is found in the middle of $P^X$. That's convenient! Denote $x_{middle}$ the median x-coordinate.

   Let $P_1$ be the set of points in the strip to the left of this line; in other words, these are the points $p \in P$ such that $x_{middle} - d < p_x < x_{middle}$. Given $P_X$ and $P_Y$, show how you can find $P_1^X$ (the points in $P_1$ sorted by their x-coordinates) and $P_1^Y$ (the points in $P_1$ sorted by their y-coordinates).

6. The divide-and-conquer algorithm is guaranteed to run in $O(n \lg n)$ time for any set of points in the plane. It is often the case that data is "nice", that is, you can make certain assumptions about the data, and exploit these assumptions to come up with simpler and more efficient algorithms.

   Assume that the set of points $P$ is uniformly distributed. Come up with a different idea to find the closest pair. Can you get $O(n)$ time?

   *Hint: throw a grid over the points. For the sake of the analysis, assume a grid of $k$-by-$k$ cells. How many points do you expect to fall in each cell, on the average? What value of $k$ would you pick ?*